Sensors and Motor Control Lab

Individual Lab Report #01

Parv Parkhiya

February 14th, 2019

Team H:
PhoeniX
Teammates:
Shubham Garg
Akshit Gandhi
Zhihao (Steve) Zhu

Individual Progress

Sensors and Motor Control:

For the sensors and motor control lab, I was assigned the job of creating the GUI (Graphical User Interface) for the system and integrating every team member's code into a common framework. I had to start working at the same time when other teammates are working on electronics or have just started writing the code. A streamlined way would have been one of the following. Either I create a framework beforehand and ask other teammates to write their individual functions based on the template or they give me their written functions and I figure out the integration part. My first target was to figure out the GUI library that I would use. One interesting solution I found was that Device Droid plugin along with SerialUI library that allows for integrating GUI specific code inside the Arduino itself. It also had data plotting API available. While most traditional GUI requires a GUI specific executable installed on the computer, Device Droid requires a common plugin and GUI specific information would be written inside the Arduino. This allows us to just connect the Arduino with our GUI code to any computer and directly use our GUI.

Arduino having the limited computing power and memory constraints, there was a certain limitation on the GUI. As a proof of concept that everything works, I started writing a dummy code with basic GUI functionality such as state button, value slider and plotting graph without any motor or sensor specific code. Getting around with the library and how various API are used inside took some time figuring out. Once I was able to pass value from GUI to Arduino and plot the value from Arduino to GUI, I started working on the actual UI for the sensors and motor control lab. This went really smooth and I was quickly able to populate to GUI frontend and backend functions template for different motors and sensors. I quickly tested the firmware without connecting any sensors and everything was working as expected. GUI can be seen in figure 1.

I provided the necessary template information to all the teammates and provided them for guidelines that would make the integration part easy. But by the time I gave them the template, some of them had already written portion of their code. After a couple of days, every teammate shared the code and all of them were quite different. I had to manually add various declaration, setup functions and main loop at appropriate places in the GUI framework. Integration was a real challenge and took a while to fix everything. I will talk more about that in the challenge section. One GUI part was properly integrated with all motor and sensors, I helped other teammates in attaching motor and sensors on the wooden plates, making holes etc. for the final presentation. For team H, I was also the presenter for the sensors and motor control lab.

Note: GUI code is attached as ZIP with submission

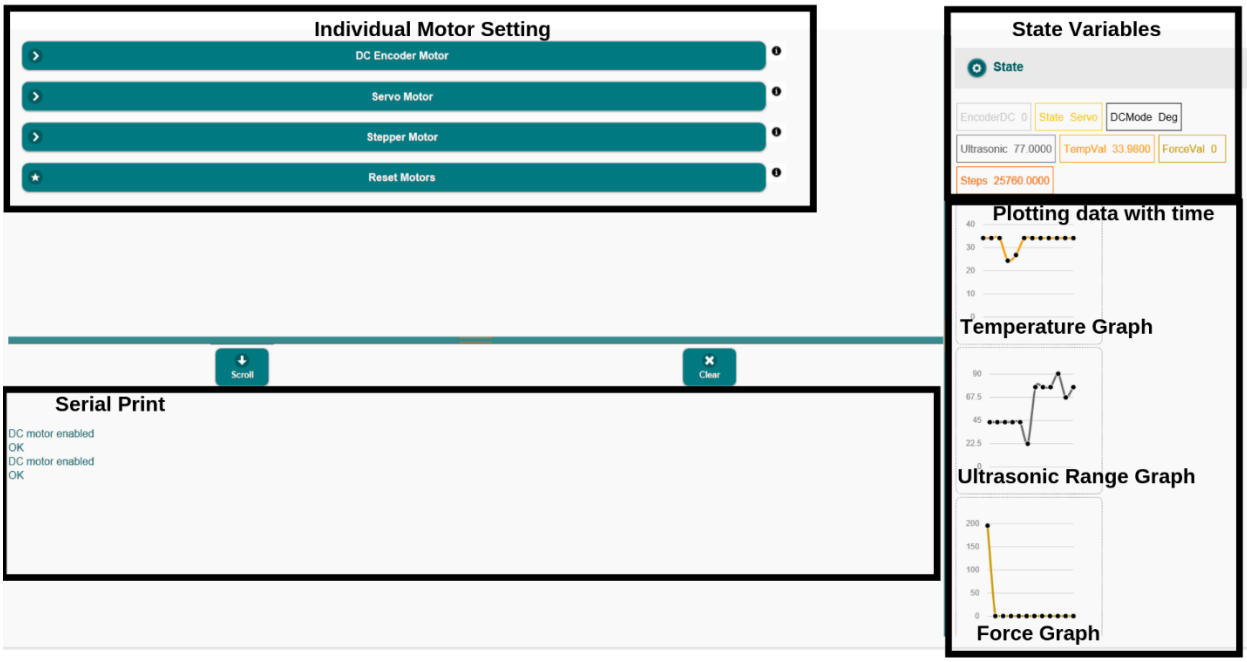

Figure 1 Main GUI Screen (Annotated)

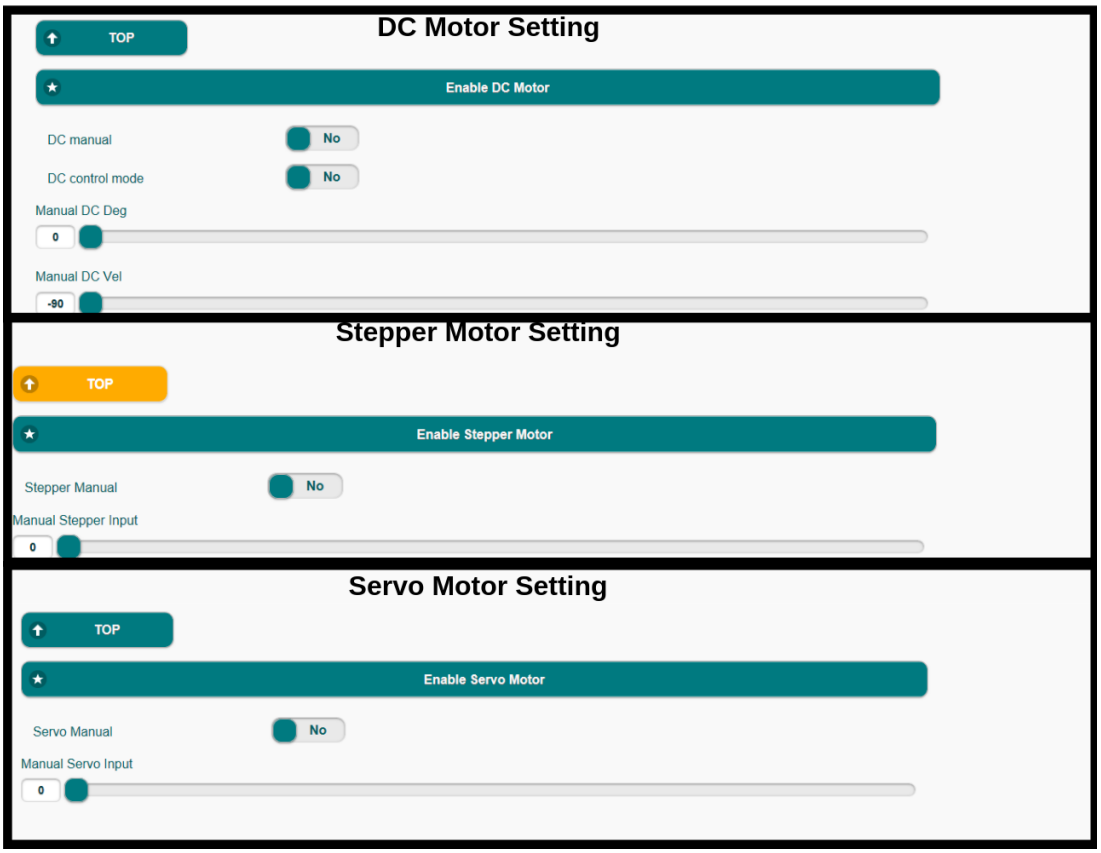

Figure 2 Various Motor Manual Setting

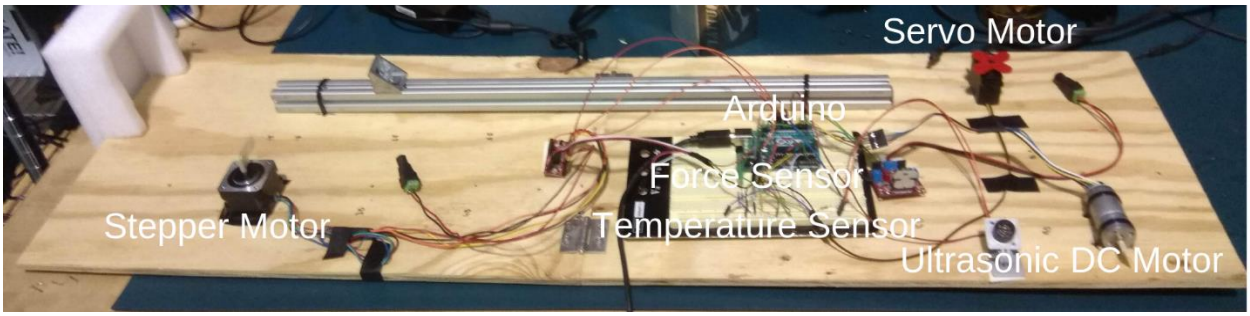

Figure 3 Final Sensors and Motor lab rig

Progress on Project:

I along with Akshit and Shubham were on campus for the most part of winter break and started working on the project very early on. There has been significant progress made since we start working on the project and giving full description is out of the scope of the ILR. A brief list of various tasks achieved is as follow. All of us together with the guidance of AirLab, made the hexacopter from scratch using various parts. We assembled the frame, added 6 motors, connected motor driver ICs, mounted pixhawk controller, added GPS module, added a battery and made necessary connections between them. We flashed the pixhawk, calibrated the drone and done multiple flight tests and made improvements to get a stable flight. After getting a stable flight, we mounted the Jetson on the drone, established the pixhawk and Jetson communication. Ubuntu, CUDA, Zed Sensors drivers, ROS were installed on the Jetson. We have also made ubuntu, Zed sensor, ROS integration on the Husky (UGV) and have been able to manually control the Husky (UGV). We also made modification on the water deploying mechanism to achieve a long range. It's important to note that all these tasks are achieved collaboratively since everyone was able to work at the same time in the lab. After the start of the semester, work has been divided so that people can work at their convenience.

Challenges

Sensors and Motor Control:

The major challenge in the sensors and motor control was regarding integration. Teammates had used overlapping pins. Pins had to be redistributed to connect all the sensors and motor to one Arduino while considering that only specific pins have an interrupt and PWM capabilities. Teammates code had a couple of same-named global variables.

Once all of that was fixed, UI was not getting sensor data reliably for some reason. After a long debugging, I figure out that issue was with returning global variable directly to SerialUI handle. Why would that create an issue is not entirely clear but probably due to the way SerialUI library works under the hood. Once that was fixed, we had a whole system working although not very impressively. Sensors and motor were connected with loose wires and connections were easily coming out. We had to reconnect wires with a more stable version and place everything on a wooden board.

Debugging the entire system was also a major challenge that required inputs from all the team members to get it resolved. That part is discussed further in the Teamwork section.

Progress on Project:

While working during the winter break, every day there was a new challenge. From installing drivers for a particular sensor or board to creating a physical mount to attach it to the hexacopter. We collectively confronted all the challenges head-on. Some took a couple of hours and some took a couple of days to figure out. Of course, the biggest challenge was getting a manual stable hexacopter flight. We crashed multiple times and even broke landing gear, GPS mount, a couple of propellers. We had to go out countless times in cold to calibrate IMU to avoid interference present in an indoor setting. Again, detailing each challenge is out of scope for this ILR.

Teamwork

Sensors and Motor Control:

My assigned task being GUI, teamwork was essential in integrating every teammates' code and even helping them debug their part when everything was integrated. I have already touched teamwork aspects while talking about challenges in this ILR.

Every sensor and motor also made minor issues such as ultrasonic sensor data were very noisy and DC encoder motor had large overshoot to name a few. While individual member had to fix just their motor and code, I had to be with all of them while they debug their part because they weren't familiar with SerialUI code and for most bugs, we didn't know whether the issue was with UI part or with their function. DC encoder motor performance was finally resolved after tuning gain values with Akshit and ultrasonic sensor result was improved using a filter implemented by Shubham.

Integration part would have been much easier if all the members had followed a specific template from the start. Nonetheless,

Project:

Because of the unique situation of winter break, everyone was usually present in the lab and we tackled the challenge together for the most part. Calibrating drone, setting up the cage for the drone testing, building drone required help from all members of the team. When some code failed to compile, everyone would start searching for the issue. While it may seem inefficient, it helped us solve any problems in a very short duration and make significant progress. Of course, such tight collaboration is not possible after the break since everyone has other assignments and deadlines. Our future plan involves the division of tasks and is included in the upcoming section.

Plan

Akshit and Shubham have taken priority in working with UAV (Hexacopter) based on their previous experience working on drones. Steve and I have taken the task of working with the UGV (Husky) ground vehicle. Our next target is to move both the vehicle autonomously in a very simple predefined way. UAV should take off, hold and land without any manual inputs and UGV should drive forward, turn by some angle and drive backward. While most of our work till now was focused on getting UAV to fly and we have just started working with UGV, we believe working with UGV should be easier compare to UAV. We are prepared to allocate more resources (work time) towards UAV if required.

For UGV (Husky), we need to mount a stereo camera and LiDAR sensor. Also, currently single board computer for UGV is Zotac PC and power distribution system that would power Zotac and LiDAR needs to figure out. We are planning to do this as part of the PCB Assignment. After that my long-term goal is to get ORB SLAM up and running first on UGV and then on UAV as well and integrate it with ROS action server framework which we have decided to use.

Quiz

Q 1: Datasheet Reading

g corresponds to gravitational acceleration on earth's surface

- Typical Sensor Range is from -3.6 g to +3.6 g (minimum range is -3 g to +3 g)
- Typical Dynamic Sensor Range is 7.2 g (minimum 6 g)
- C_{dc} acts as a low pass filter to counter the voltage fluctuation present in the power supply. Capacitor achieves this by resisting the rapid change in voltage ($i = C\frac{dV}{dt}$) by providing or absorbing additional charge.
- Transfer Function $V_{out} = 1.5V + \frac{300mV}{g}a$ (where a is acceleration input)
- Largest nonlinearity error: $\pm 0.3\%\ of\ Full\ Scale$ i.e. $\pm 3.6g * \frac{0.3}{100} = \pm 0.0108g$
- For both X and Y axis, noise density is $150\mu g/\sqrt{Hz}$

$$\therefore RMS\ Noise\ at\ 25\ Hz = 150 * \sqrt{25 * 1.6} * 10^{-6}g = 0.000948g$$

- To experimentally determine the Noise at 0 Hz, we would need to measure the reading at constant acceleration. The simplest way to get a constant acceleration is free fall. Sensor along with a microcontroller that can read the analog value and store the value can be dropped to free fall in Vacuum tube. Now the measured a and actual g can be compared to get the noise information at 0 Hz.

Q 2: Signal Conditioning

Filtering

- Moving Average Filter:
  - While moving average filter does remove high-frequency noise to some extent but it also causes loss of information especially by blurring the strong edges present in the image.
  - Deciding the size of the window is challenging and can lead to a better or worse result.
- Median Filter:
  - Unlike moving average filter, Median filter assigns pixel value from the surrounding neighbor pixel values which lead to strong edges being preserved. But the smoothing and noise removal is not as good as moving average filter.
  - Deciding the size of the window is also challenging here like Moving Average filter.

- OpAmp:
  - For the given standard circuit, the equation can be written as

$$V_{out} = \left(1 + \frac{R_f}{R_i}\right)V_2 - \frac{R_f}{R_i}V_1$$

Sensor 1: Uncalibrated sensor has a range of -1.5 to 1.0V (-1.5V should give a 0V output and 1.0V should give a 5V output)
Let $V_1 = V_{ref}$ then OpAmp equation can be written as follow,

$$0 = \left(1 + \frac{R_f}{R_i}\right)(-1.5) - \frac{R_f}{R_i}V_{ref}$$

$$5 = \left(1 + \frac{R_f}{R_i}\right)(1) - \frac{R_f}{R_i}V_{ref}$$

Subtracting above equation,

$$5 = 2.5(1 + \frac{R_f}{R_i})$$

$$\therefore \frac{R_f}{R_i} = 1$$

Putting it back in equation yields,

$$5 = (1 + 1)(1) - (1)V_{ref}$$

$$\therefore V_{ref} = -3$$

Transfer function can be given by,

$$V_{out} = 2 * V_2 + 3$$

Thus, our assumption is correct. V_1 is the reference voltage and its value is -3. V_2 is the input voltage. Also, $\frac{R_f}{R_i} = 1$

Sensor 2: Uncalibrated sensor has a range of -2.5 to 2.5V (-2.5V should give a 0V output and 2.5V should give a 5V output)
Let $V_1 = V_{ref}$ then OpAmp equation can be written as follow,

$$0 = \left(1 + \frac{R_f}{R_i}\right)(-2.5) - \frac{R_f}{R_i}V_{ref}$$

$$5 = \left(1 + \frac{R_f}{R_i}\right)(2.5) - \frac{R_f}{R_i}V_{ref}$$

Subtracting above equation,

$$5 = 5(1 + \frac{R_f}{R_i})$$

$$\therefore \frac{R_f}{R_i} = 0$$

Putting it back in the equation,

$$5 = (1 + 0) * 2.5 - 0 * V_{ref}$$

$$5 = 2.5$$

Therefore, our assumption is false.

Let $V_2 = V_{ref}$ then OpAmp equation can be written as follow,

$$0 = \left(1 + \frac{R_f}{R_i}\right) V_{ref} + \frac{R_f}{R_i} 2.5$$

$$5 = \left(1 + \frac{R_f}{R_i}\right) V_{ref} - \frac{R_f}{R_i} 2.5$$

Subtracting above equation,

$$5 = -5(\frac{R_f}{R_i})$$

$$\therefore \frac{R_f}{R_i} = -1$$

But $R_f \; and \; R_i$ are resistors and can not have negative resistance value.
Therefore, this assumption is false as well.
Thus, no value of R_f, R_i or Selection of V_1, V_2 as reference voltage can lead to desired output mapping.

Q 3: Control

- PID Control: Simplest way would be to construct an error term as follow:

$$e_t = \theta_t^{desired} - \theta_t^{actual}$$

Where θ_t^{actual} will be θ_t^{sensed} using some encoder or other sensor.

Derivative Term:

$$\dot{e_t} = \dot{\theta}_t^{desired} - \dot{\theta}_t^{actual}$$

Now, if just want to reach a position ($no \; \dot{\theta}_t^{desired}$) in which case $\dot{e}$ can be approximated as follow

$$\dot{e_t} = \frac{e_t - e_{t-\Delta t}}{\Delta t}$$

Integral Term:
Similarly, the integral term can be approximated as follow,

$$\int_0^t e \; dt \approx e_{accumulated(t)} \approx e_{accumulated(t-\Delta t)} + e_t * \Delta t$$

Finally, the control input acceleration or the PWM can be given by,

$$input \; u_t = K_p * e_t + K_d * \dot{e_t} + K_i * \int_0^t e \; dt$$

- To remove, sluggish behavior, I would increase proportional gain K_p which will increase the net input to the system for quicker convergence. Since K_p directly corresponds to the error present in the system.
- To remove the steady-state error, I would increase the integral gain K_i since, in case of a steady error that integral term would keep adding the steady state error in input and would over time counter the error when this integral term becomes a sufficiently large value.

To reduce the overshoot, I would increase the derivative gain K_d since it would keep a check on the rate of change so that in an attempt to reach the goal location, we don't end up with a high velocity which causes the overshoot when we reach goal location.